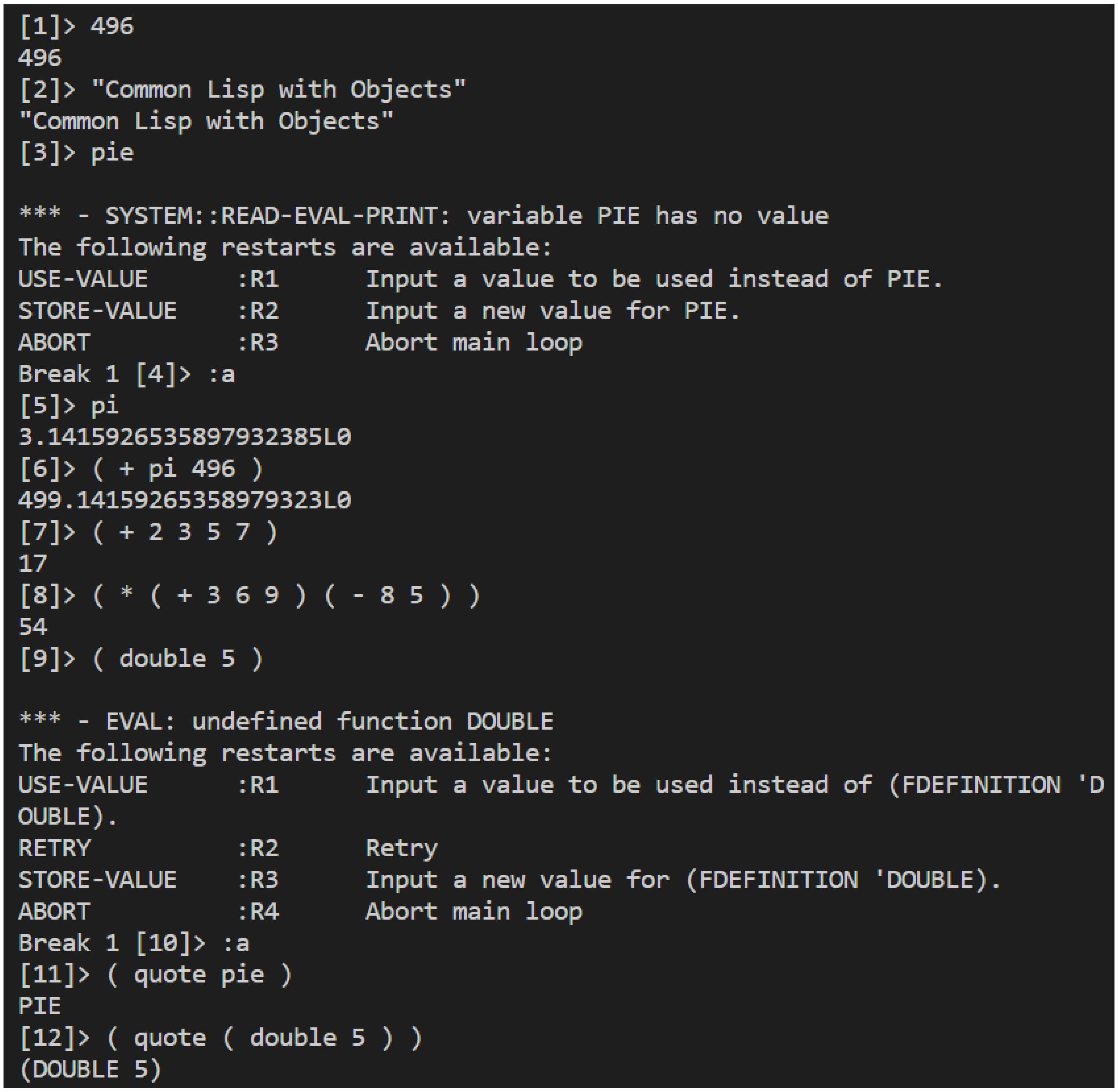
AI Assignment: First Lisp Interactions and Problem Solving

Abstract:

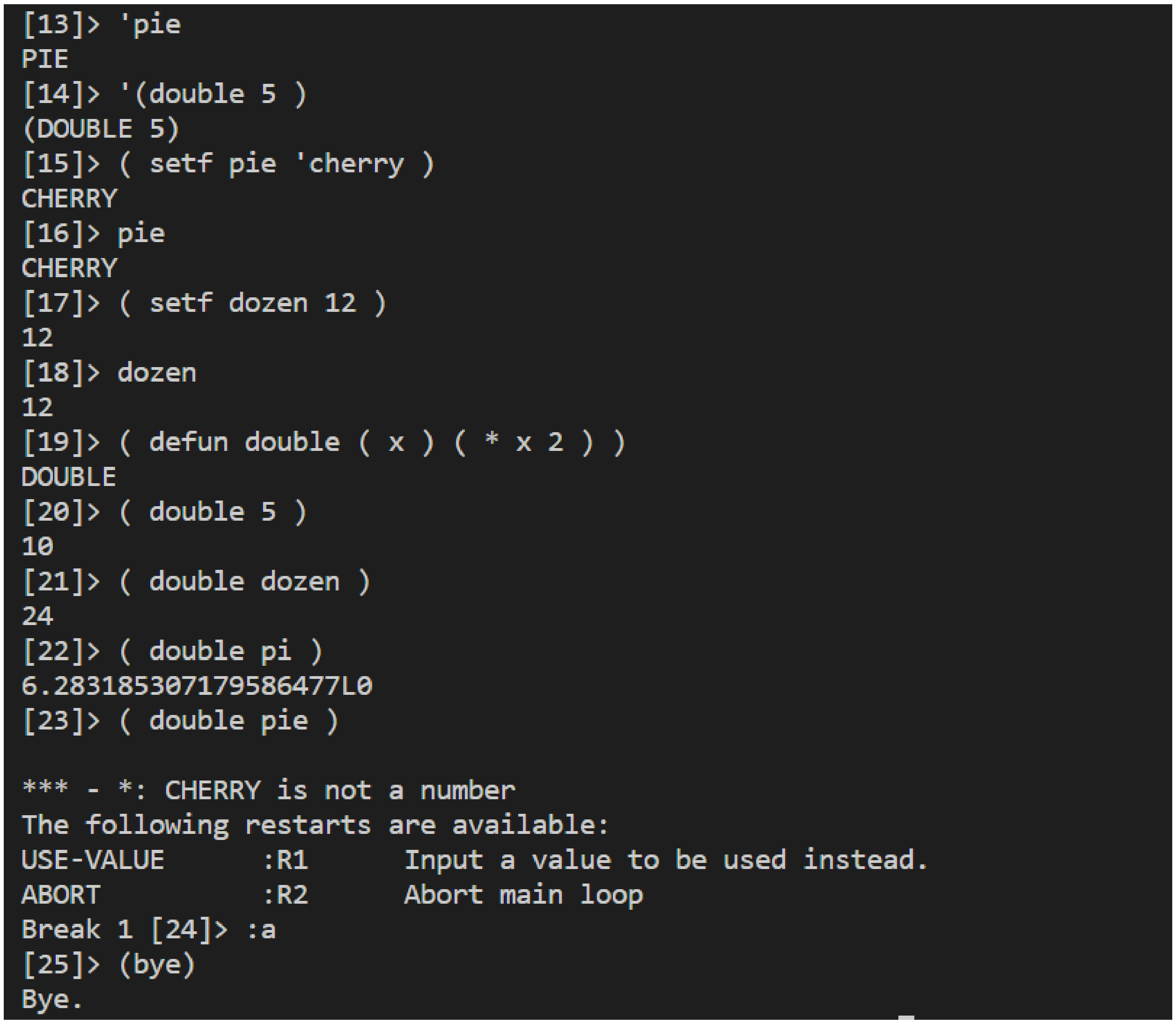
This assignment is a basic introduction to Common Lisp. The first two assignments are a reproduction of a couple of demos and the other two problems are more mentally demanding. The first of them involves constructing a solution for measuring the percentage of white space on a dice and the second involves a goat's grazing area.

Task 1: Basic Form's Demonstration



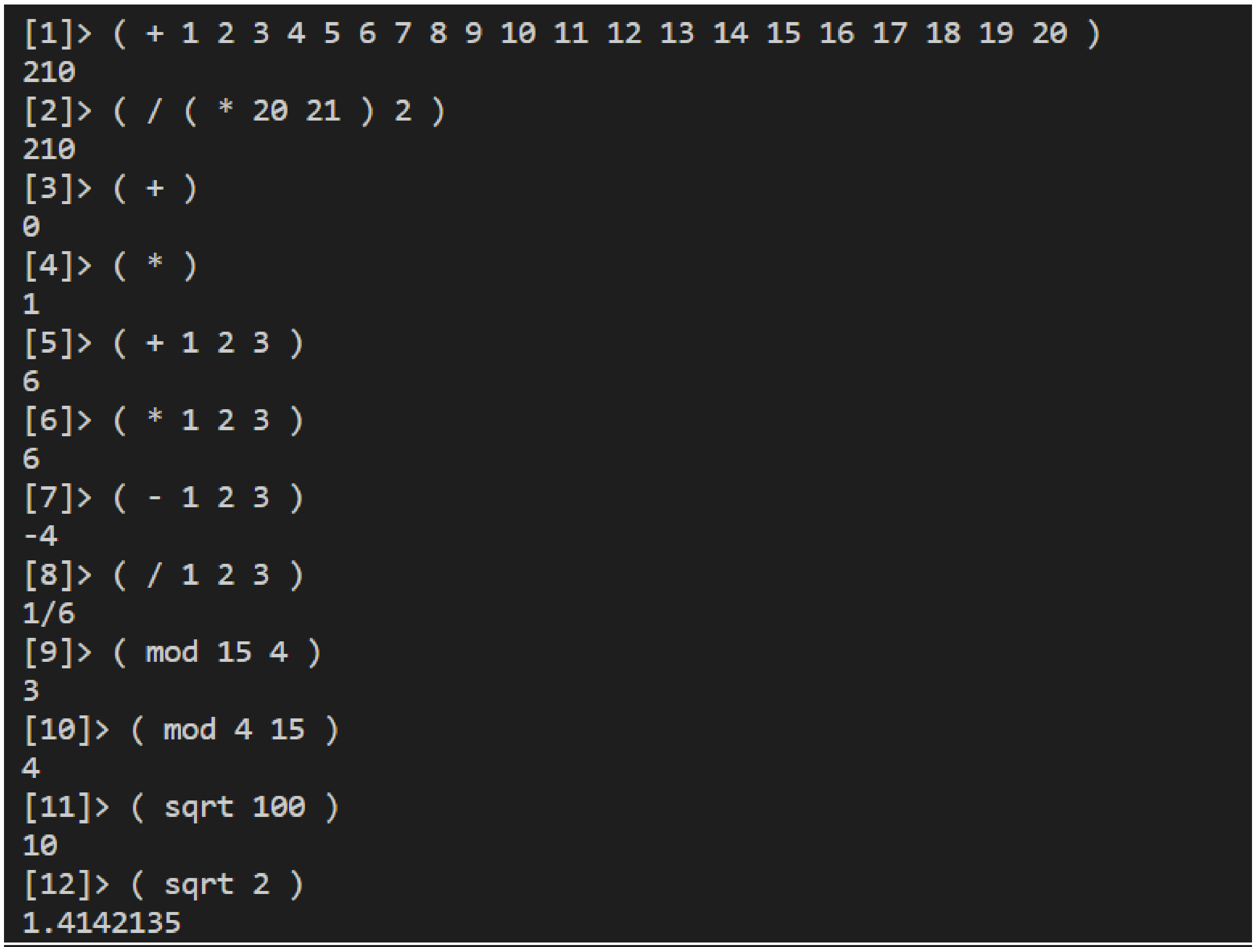
```
[1]> 496
496
[2]> "Common Lisp with Objects"
"Common Lisp with Objects"
[3]> pie

*** - SYSTEM::READ-EVAL-PRINT: variable PIE has no value
The following restarts are available:
USE-VALUE      :R1      Input a value to be used instead of PIE.
STORE-VALUE    :R2      Input a new value for PIE.
ABORT          :R3      Abort main loop
Break 1 [4]> :a
[5]> pi
3.1415926535897932385L0
[6]> ( + pi 496 )
499.14159265358979323L0
[7]> ( + 2 3 5 7 )
17
[8]> ( * ( + 3 6 9 ) ( - 8 5 ) )
54
[9]> ( double 5 )

*** - EVAL: undefined function DOUBLE
The following restarts are available:
USE-VALUE      :R1      Input a value to be used instead of (FDEFINITION 'D
OUBLE).
RETRY          :R2      Retry
STORE-VALUE    :R3      Input a new value for (FDEFINITION 'DOUBLE).
ABORT          :R4      Abort main loop
Break 1 [10]> :a
[11]> ( quote pie )
PIE
[12]> ( quote ( double 5 ) )
(DOUBLE 5)
```

Task 1: Basic Form’s Demonstration Cont…

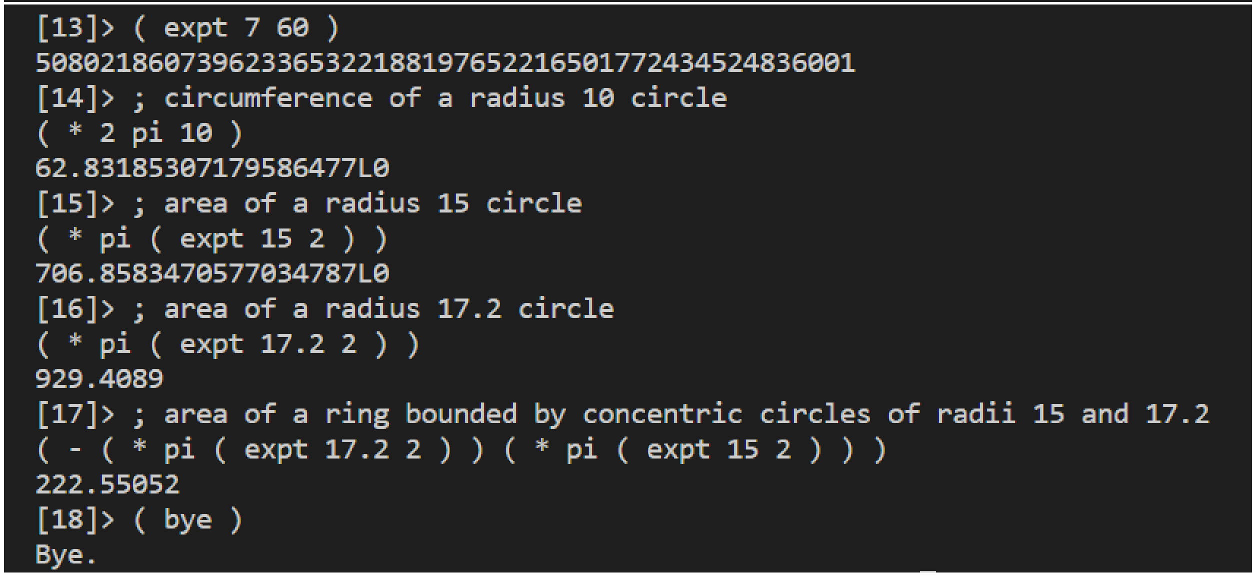


```
[13]> 'pie
PIE
[14]> '(double 5 )
(DOUBLE 5)
[15]> ( setf pie 'cherry )
CHERRY
[16]> pie
CHERRY
[17]> ( setf dozen 12 )
12
[18]> dozen
12
[19]> ( defun double ( x ) ( * x 2 ) )
DOUBLE
[20]> ( double 5 )
10
[21]> ( double dozen )
24
[22]> ( double pi )
6.283185307179586477L0
[23]> ( double pie )

*** - *: CHERRY is not a number
The following restarts are available:
USE-VALUE      :R1      Input a value to be used instead.
ABORT          :R2      Abort main loop
Break 1 [24]> :a
[25]> (bye)
Bye.
```

Task 2: Numeric Forms Demonstration



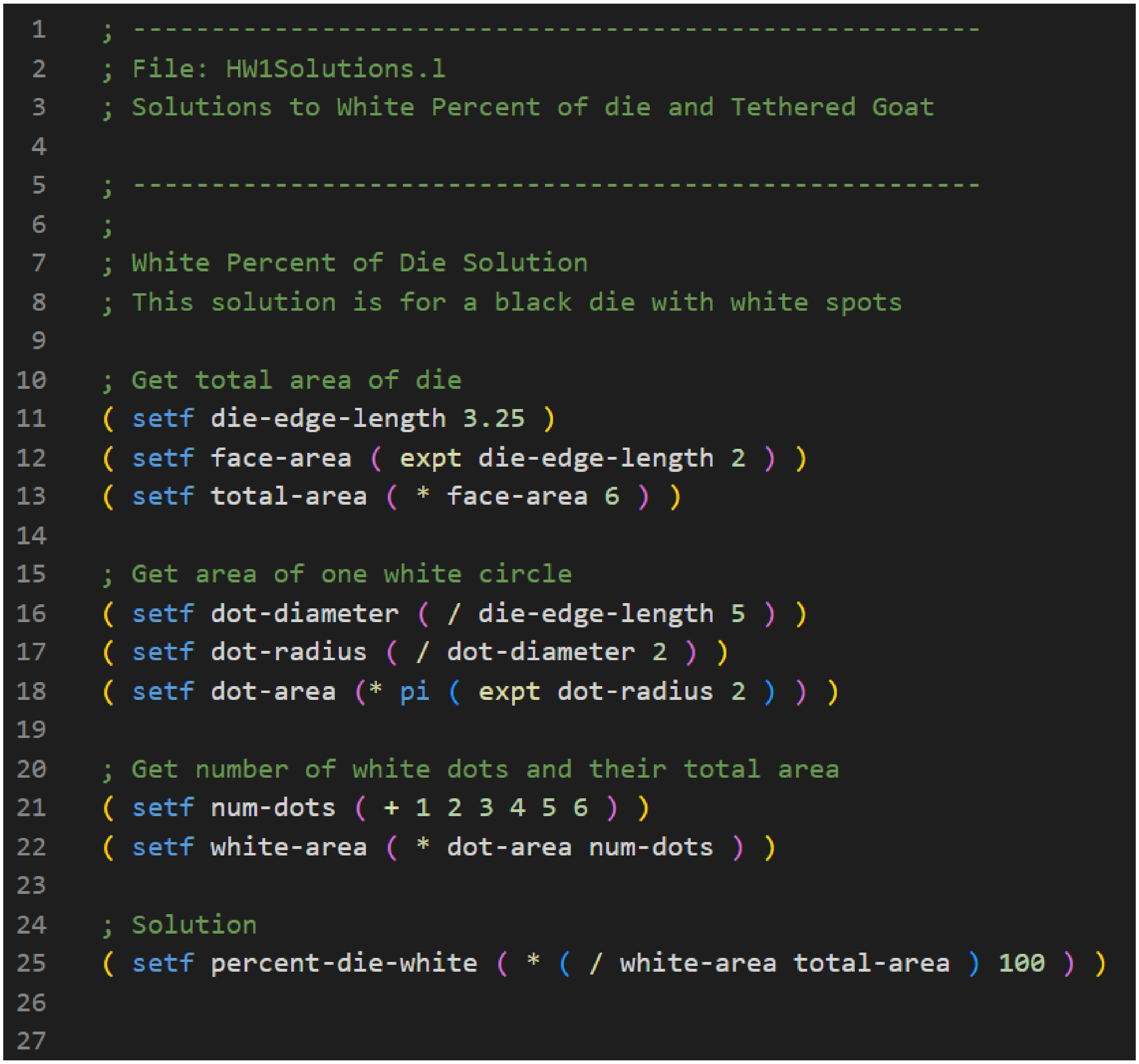
```
[1]> ( + 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 )
210
[2]> ( / ( * 20 21 ) 2 )
210
[3]> ( + )
0
[4]> ( * )
1
[5]> ( + 1 2 3 )
6
[6]> ( * 1 2 3 )
6
[7]> ( - 1 2 3 )
-4
[8]> ( / 1 2 3 )
1/6
[9]> ( mod 15 4 )
3
[10]> ( mod 4 15 )
4
[11]> ( sqrt 100 )
10
[12]> ( sqrt 2 )
1.4142135
```



```
[13]> ( expt 7 60 )
508021860739623365322188197652216501772434524836001
[14]> ; circumference of a radius 10 circle
( * 2 pi 10 )
62.83185307179586477L0
[15]> ; area of a radius 15 circle
( * pi ( expt 15 2 ) )
706.8583470577034787L0
[16]> ; area of a radius 17.2 circle
( * pi ( expt 17.2 2 ) )
929.4089
[17]> ; area of a ring bounded by concentric circles of radii 15 and 17.2
( - ( * pi ( expt 17.2 2 ) ) ( * pi ( expt 15 2 ) ) )
222.55052
[18]> ( bye )
Bye.
```

Task 3: “Percent of Die that is White” problem

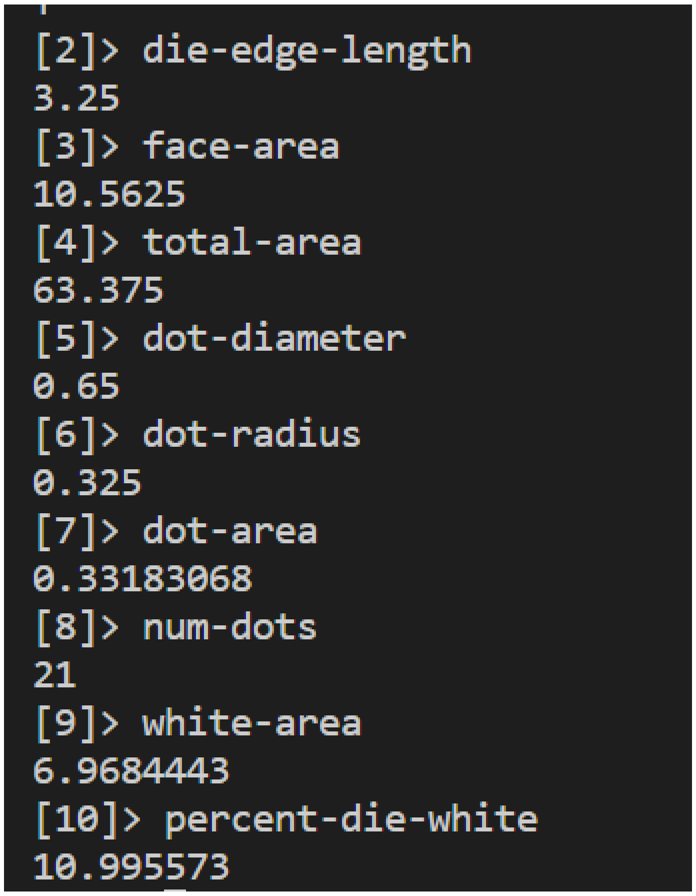
My solution to this problem is based on a standard black die with white spots.



```
; -----------------------------------------------------
; File: HW1Solutions.l
; Solutions to White Percent of die and Tethered Goat

; -----------------------------------------------------
;
; White Percent of Die Solution
; This solution is for a black die with white spots

; Get total area of die
( setf die-edge-length 3.25 )
( setf face-area ( expt die-edge-length 2 ) )
( setf total-area ( * face-area 6 ) )

; Get area of one white circle
( setf dot-diameter ( / die-edge-length 5 ) )
( setf dot-radius ( / dot-diameter 2 ) )
( setf dot-area (* pi ( expt dot-radius 2 ) ) )

; Get number of white dots and their total area
( setf num-dots ( + 1 2 3 4 5 6 ) )
( setf white-area ( * dot-area num-dots ) )

; Solution
( setf percent-die-white ( * ( / white-area total-area ) 100 ) )


```

Task 3: “Percent of Die that is White” problem Cont…

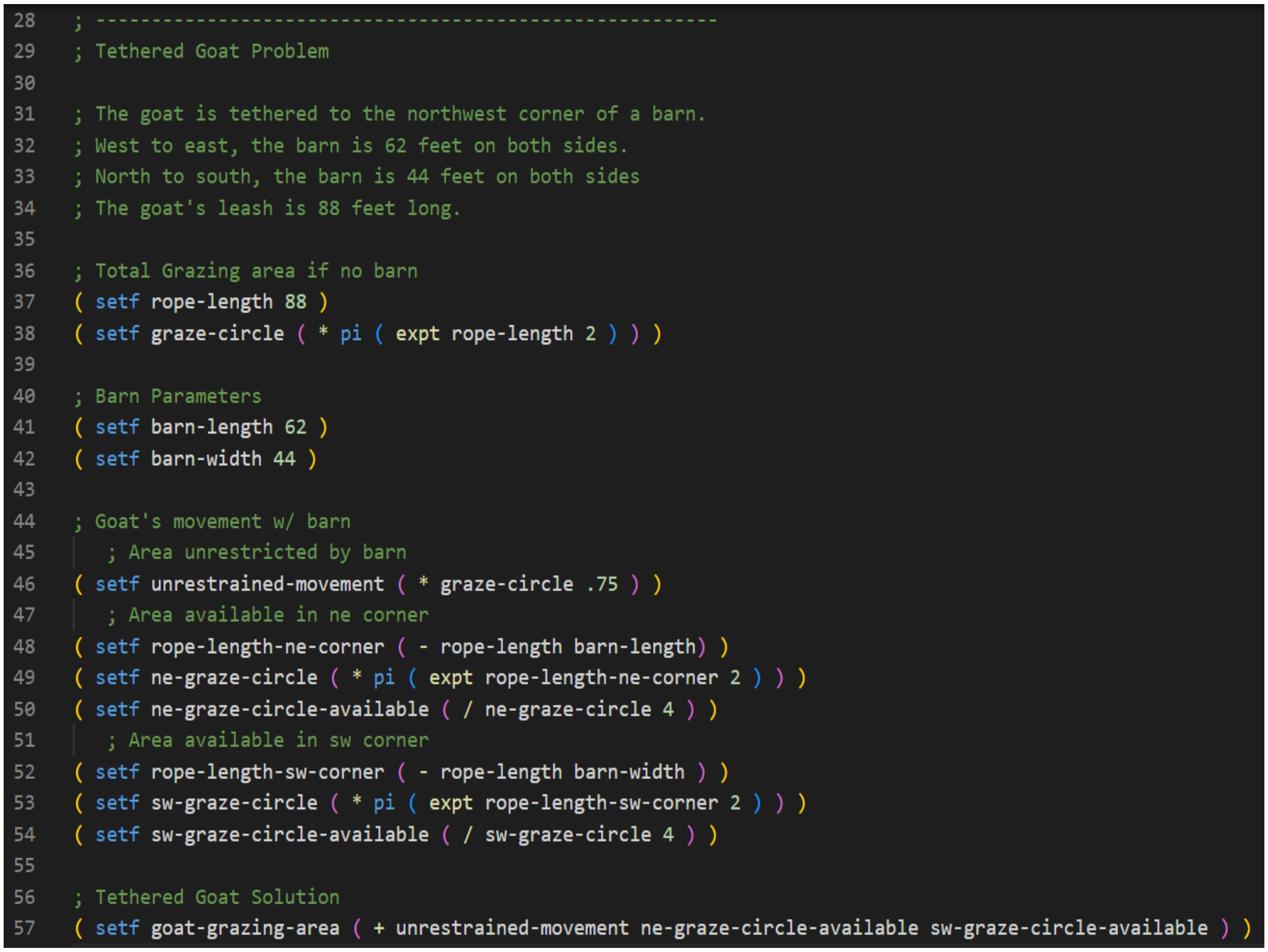
These are the common Lisp responses to the objects created in the Lisp file.



```
[2]> die-edge-length
3.25
[3]> face-area
10.5625
[4]> total-area
63.375
[5]> dot-diameter
0.65
[6]> dot-radius
0.325
[7]> dot-area
0.33183068
[8]> num-dots
21
[9]> white-area
6.9684443
[10]> percent-die-white
10.995573
```

Task 4: “Tethered Goat” problem

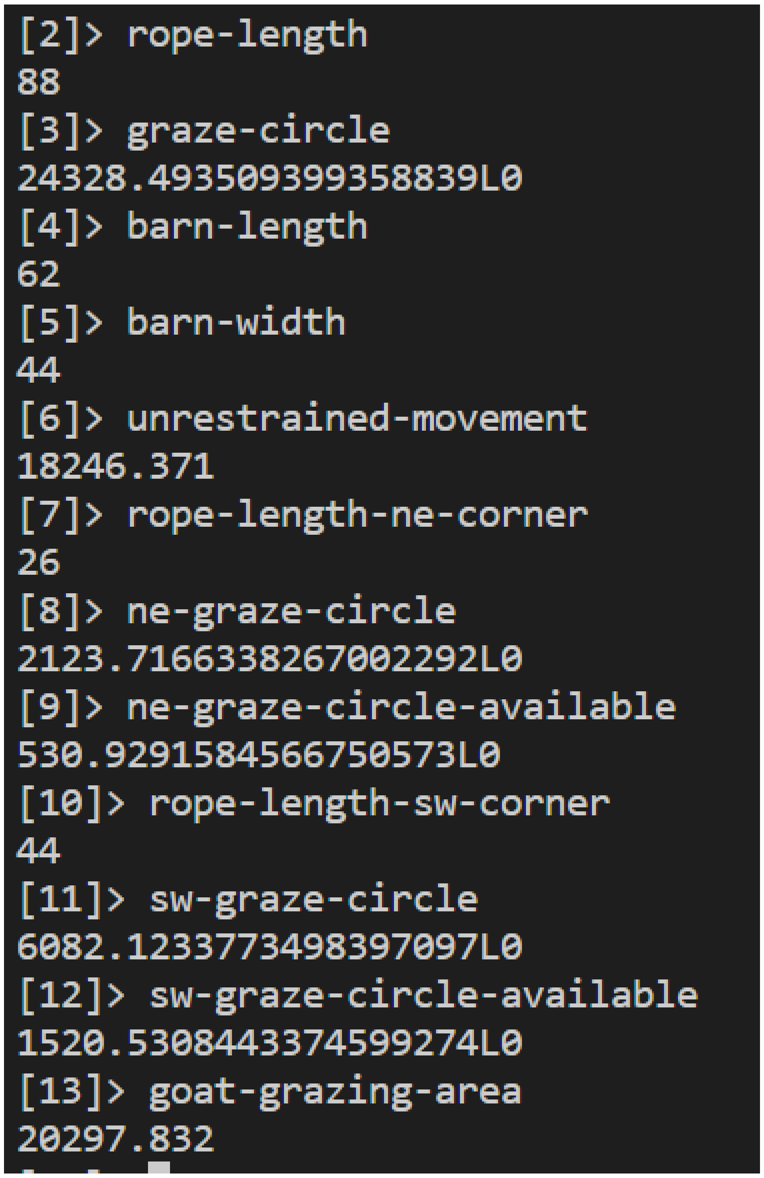
No goats were harmed in the production of this solution.



```lisp
; --------------------------------------------------------
; Tethered Goat Problem

; The goat is tethered to the northwest corner of a barn.
; West to east, the barn is 62 feet on both sides.
; North to south, the barn is 44 feet on both sides
; The goat's leash is 88 feet long.

; Total Grazing area if no barn
( setf rope-length 88 )
( setf graze-circle ( * pi ( expt rope-length 2 ) ) )

; Barn Parameters
( setf barn-length 62 )
( setf barn-width 44 )

; Goat's movement w/ barn
   ; Area unrestricted by barn
( setf unrestrained-movement ( * graze-circle .75 ) )
   ; Area available in ne corner
( setf rope-length-ne-corner ( - rope-length barn-length) )
( setf ne-graze-circle ( * pi ( expt rope-length-ne-corner 2 ) ) )
( setf ne-graze-circle-available ( / ne-graze-circle 4 ) )
   ; Area available in sw corner
( setf rope-length-sw-corner ( - rope-length barn-width ) )
( setf sw-graze-circle ( * pi ( expt rope-length-sw-corner 2 ) ) )
( setf sw-graze-circle-available ( / sw-graze-circle 4 ) )

; Tethered Goat Solution
( setf goat-grazing-area ( + unrestrained-movement ne-graze-circle-available sw-graze-circle-available ) )
```

Task 4: “Tethered Goat” problem Cont…

These are the common Lisp responses to the objects created in the Lisp file for the tethered goat.



```
[2]> rope-length
88
[3]> graze-circle
24328.493509399358839L0
[4]> barn-length
62
[5]> barn-width
44
[6]> unrestrained-movement
18246.371
[7]> rope-length-ne-corner
26
[8]> ne-graze-circle
2123.7166338267002292L0
[9]> ne-graze-circle-available
530.9291584566750573L0
[10]> rope-length-sw-corner
44
[11]> sw-graze-circle
6082.1233773498397097L0
[12]> sw-graze-circle-available
1520.5308443374599274L0
[13]> goat-grazing-area
20297.832
```